Scaling GPA* for complex protein folding pathway simulations

by

Foram Patel

A thesis presented to the Honors College of Middle Tennessee State University in partial fulfillment of the requirements for graduation from the University Honors College

Spring 2023

Thesis Committee:

Dr. Joshua Phillips, Thesis Director

Dr. Philip E. Phillips, Committee Chair

Scaling GPA* for complex protein folding pathway simulations

by

Foram Patel

APPROVED:

_____

Dr. Joshua Phillips, Thesis Director

Associate Professor, Department of Computer Science

_____

Dr. Philip E. Phillips, Thesis Committee Chair

Associate Dean, University Honors College

Acknowledgments

I would like to thank the University Honors College for allowing me to grow academically and helping me to conquer my fears. I am exceptionally thankful to Dr. Joshua Phillips, Assistant Professor of Computer Science, my research advisor, and mentor, for his valuable advice throughout my undergraduate education journey and for guiding me through this rewarding experience. I also want to thank the Office of Research and Sponsored Programs at Middle Tennessee State University for their support. Lastly, I would like to thank my parents for their support in my career ambitions and for their encouragement and love.

Abstract

Finding improved protein folding pathway modeling tools is crucial to develop more potent treatments for disorders caused by protein misfolding. The fast-folding streptococcal protein G (1GB1), which has alpha-helices and several beta-sheets, can be used to assess models of protein folding. Pathway prediction is often computationally expensive and time-consuming, so current research focuses on accelerating Molecular Dynamics (MD) simulations. To fix the issue of proteins getting trapped in minima, past methods have imposed an unnatural bias on the potential and kinetic energies of the simulation environments. Finding unbiased methods for MD simulations was an open problem and was addressed by (Syzonenko & Phillips, 2020), introducing a combination of the A* algorithm and MD simulations. The current implementation has storage issues due to an abundant number of files produced preventing large-scale implementation. A viable alternative could be the replacement of auxiliary file storage on disk with a key-value data structure for storage. This would prove less burdensome on the file systems. Instead of relying on GROMACS commands using OS system calls, the MDAnalysis library, which is based on GROMACS, may be used for simulation commands and storing coordinates. Once validated on the complex and fast-folding 1GB1 protein, the approach may be applied to even larger α-β proteins.

Table of Contents

List of Figures

Chapter 1 Introduction

1.1. Proteins

Proteins are big, complex molecules that serve several important roles in the body. These include digesting, transportation, structural functions, defense, storage, and mobility. Proteins are important in the movement of chemicals throughout the body. Hemoglobin, which transports oxygen throughout the body, is one such example. Proteins such as Immunoglobulin also play a vital role in the immune system, protecting the body against infections. Each protein in a cell has a distinct function. This is enabled by a basic characteristic of proteins: they fold. Protein folding refers to the ongoing and universal process by which the long, coiled strings of amino acids that comprise proteins in all living things fold into more sophisticated three-dimensional structures. (McGill University, 2011).

1.2. Protein Folding

A protein is made up of a lengthy chain of amino acids. The arrangement of these amino acids controls how the protein chain folds and how it functions. Some parts of the protein chain coil up into the highly common slinky-like forms known as alpha-helices or zigzag patterns known as beta-sheets (which resemble the folds of a paper fan). These two highly common structures can interact with one another to produce more complicated structures. Proteins, despite being made up of the same fundamental building components, can perform vastly distinct roles by folding into various shapes or conformations (Geiler, 2014).

The structure of a protein is closely related to its function. For example, antibody proteins fold into forms that allow them to accurately detect and target certain foreign substances, analogous to how a key fits into a lock. Understanding the structure that proteins will fold into is thus critical to understanding how organisms work and how life functions (Toews, 2021).

1.3. Protein Folding Pathways

The Protein Folding Problem has three key issues: 1) determining the folding code, 2) how the protein folds, and 3) if it is even feasible to predict how a protein will fold. This thesis focuses on the second issue of predicting the specific pathway for protein folding. Current researchers in protein folding can predict folded protein structures with high accuracy. The protein folding pathway prediction issue aims to provide an accurate path outlining all stages taken in the process. The complete pathways are necessary for biological research because certain mutations could be added to the simulated folding states, and we could observe the resulting protein.

Knowledge of folding pathways is significant because improperly folded proteins in people cause illnesses such as Alzheimer's, Parkinson's, Huntington's, Emphysema, and Cystic fibrosis. It is critical to find better protein folding modeling techniques to create more effective treatments for such diseases. (McGill University, 2011). This can also be useful in researching methods to degenerate or disintegrate certain folding proteins. The complete states and we can observe the resulting protein. This would also be beneficial in studying ways to degenerate or disintegrate certain viral proteins, for example.

Chapter 2 Background

2.1 Molecular Dynamics (MD) Simulations

It is difficult to experimentally study such protein misfolding in these situations. For such studies, Molecular Dynamics (MD) simulations, which are physics-based models, have become quite common and effective in studying such protein folding mechanisms so far. For computational path-finding algorithms to work, an accurate representation of the 'goal state,' or final protein conformation (structure) is needed. Also, pathway prediction is often computationally expensive and time-consuming, so current research focuses on creating efficient methods that would save computing power and overall calculation times. These methods have been helpful but are still expensive in terms of computing power and overall time required. Therefore, current studies aim at speeding up these MD simulations.

Computational protein folding simulations may be used to evaluate experimental folding data, plan innovative folding studies, and investigate the impact of mutations and small molecules on folding. However, while significant experimental and computational progress has been made in understanding how tiny proteins fold, research on bigger, multidomain proteins, which make up most proteins, has underperformed. Folding simulations based on MD and native structure knowledge can give essential, comprehensive information on folding free energy landscapes, intermediates, and routes.

Furthermore, developments in computer power and methodological breakthroughs have made massive protein folding simulations practicable and beneficial (Gershenson et al., 2020). Beyond the tiny fast-folding model proteins that have historically been chosen by protein-folding researchers, research beyond the small fast-folding model proteins is required in the field of drug development and biological investigations. Protein folding presents a puzzle of such intrigue and depth that the scientific community cannot even agree on a single statement of the problem (Ben-Naim, 2012).

Some might argue that the problem has been solved in theory, but in practice, there is no simple solution. Improving molecular dynamics is one key strategy to address the protein-folding challenge. Finding a stable folded conformation for a protein is insufficient; the pathway is also a key component. One of the common issues in protein folding simulations is the protein getting trapped in local minima of energy surface states (gets stuck in the minima). It remains stuck there until the kinetic energy can push it out of this area. Revisitation of previous minima also often slows progress. The two approaches used in the past to overcome this issue are Metadynamics (Pfaendtner, 2019) and replica exchange (Zhou, 2022).

2.1.1 Replica-Exchange Molecular Dynamics (REMD)

By combining MD simulations and the Monte Carlo methodology, the REMD method is a hybrid approach. REMD simulations use MD simulations at various temperatures or the same temperature but with various Hamiltonians to simulate multiple copies (replicas) of the same system concurrently. With a probability determined by the Metropolis criterion, swapping between nearby copies is tried on occasion.

Using this technique, a generalized ensemble of the simulated system is produced. This enables investigation of the free energy landscape of protein aggregates since REMD can quickly cross high-energy barriers and sample conformational space sufficiently (Qi et al., 2017).

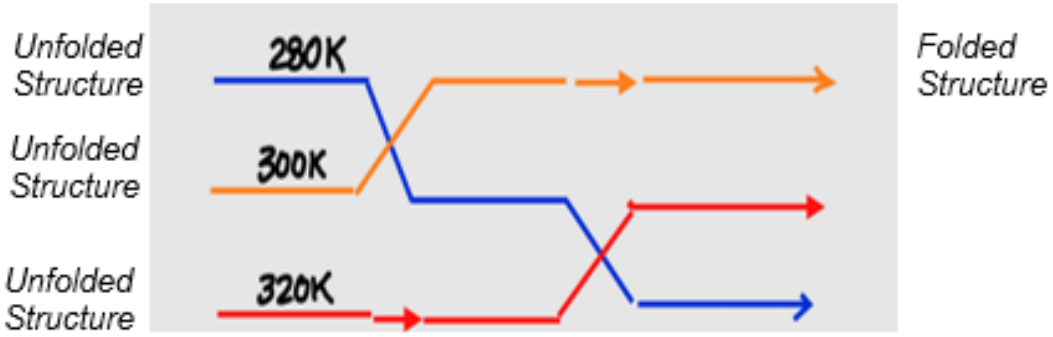

Figure 1 Replica-Exchange MD

2.1.2 Metadynamics

Metadynamics is a computational simulation technique used in molecular dynamics simulations to overcome the problem of free energy barriers and accelerate the sampling of rare events. It achieves this by adding a bias potential that increases over time, allowing the system to explore new regions of phase space. The method is particularly useful for studying complex molecular systems, such as proteins, and has applications in drug discovery, material science, and biochemistry (Lombardi & Nigro, 2018).

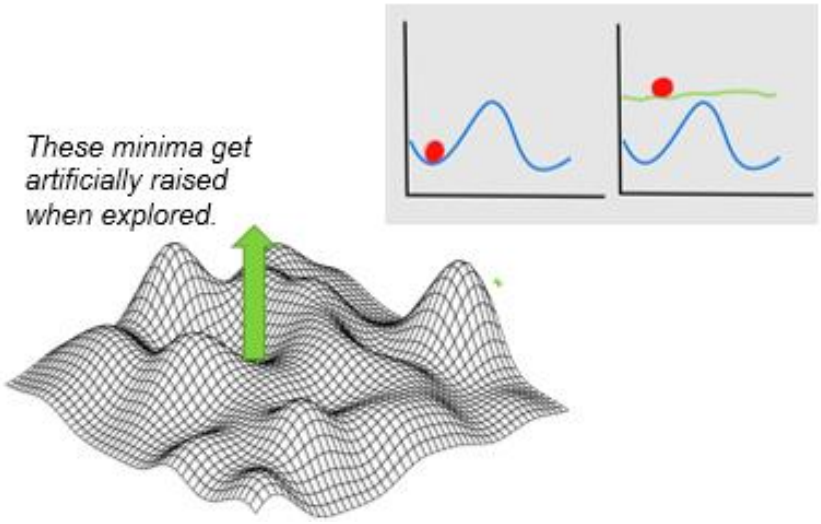

Figure 2 Metadynamics

2.1.3 Finding Unbiased Methods

However, these methods impose an *unnatural* bias on the potential and kinetic energies, respectively, of the environment of the simulations. REMD affects the kinetic energy of the system. This happens because of the elevated temperatures required to explore more states. In Metadynamics, the potential energy is affected, as this modification could potentially lead to unnatural states being visited in some other minima. Figure 3 shows graphically how the potential energy varies throughout the simulation states. It shows the probability of each state being visited, and we can see there are many peaks and valleys. While getting out of one minima, it may get stuck in another one again.

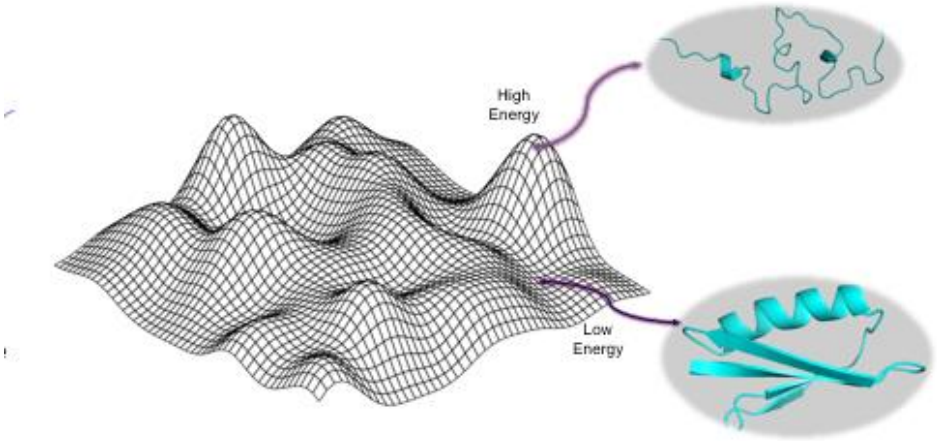

Figure 3 Potential Energy Surface

Finding *unbiased* methods for such simulations was an open problem and was addressed by (Syzonenko & Phillips, 2020), by getting rid of such unnatural biases by introducing a combination of the A* (a path-finding algorithm used in artificial intelligence) algorithm and MD simulations, as described later.

2.2 Pathfinding and Search Algorithms

To determine the shortest route between two locations in a graph, pathfinding algorithms are used. A popular pathfinding algorithm that determines the shortest route between two points in a graph with positive edge weights is called Dijkstra's algorithm. It operates by keeping track of a priority queue of nodes, choosing the node with the lowest cost to investigate next at each step (Cormen et al., 2009). Uniform-cost search is a graph traversal algorithm that seeks to discover the most cost-effective path between two nodes. It works by traversing the graph, starting from the node with the lowest cost path, giving priority to paths with the lowest accumulated cost thus far. The advantage of this algorithm is that it is guaranteed to find the optimal solution if one exists.

Conversely, greedy search is a heuristic-based algorithm that identifies the most favorable decision at each step based on a locally optimal choice. Unlike uniform-cost search, it only considers the next step leading to the target node and does not consider the entire path cost. As a result, it can provide faster solutions, but there is no guarantee that it will find the optimal solution (Russell & Norvig, 2010).

2.2.1 The A* Algorithm

The A* algorithm uses a heuristic function to guide the search toward the goal node, and it is based on the idea of combining the advantages of both Dijkstra's algorithm and the greedy Search algorithm (Hart, Nilsson, & Raphael, 1968). The A* algorithm is a combination of uniform-cost search and greedy search algorithms.

Implementation of the A* algorithm is related to the theoretical best-first search algorithm but differs in the cost estimation process i.e., $f(n) = g(n) + h(n)$, where $g(n)$ is the actual cost from the initial state to the current state and $h(n)$ is the estimates cost from current state to a goal state. Therefore, $f(n)$ is an approximate minimum distance estimate of the path of any solution going through node n. At every step in the search process, the lowest $f(n)$ is chosen for node n. The algorithm terminates when a goal state is found (Sharma & Kumar, 2016).

Syzonenko & Phillips (2020) obtained the $h(n)$ values by calculating distances between the current and folded state of the protein. They used a greedy proximal approach; we will trust the $h(n)$ values more than the $g(n)$ values. By minimizing the importance of $g(n)$, this approach encourages progress that would otherwise be stifled due to the stochastic nature of MD simulations. We can avoid the issue of getting stuck in local minima. They tested their method of integrating the A* algorithm with MD simulations (GPA*) on the following proteins: (a) the Trp-cage mini protein construct TC5b (1L2Y), which is a short, fast-folding protein with an α -helical secondary structure; (b) the immunoglobulin binding domain of the streptococcal protein G (1GB1), which contains an α-helix and several β-sheets; and (c) the chicken villin subdomain HP-35, N68H protein (1YRF), which forms from several α-helices. They compared the

technique to replica-exchange MD (mentioned above), which is the most used algorithm for speeding protein folding using MD. They discovered that GPA* not only lowered the computing time required to acquire the folded conformation without adding artificial energy bias but also enabled them to design trajectories with minimum movements required for the folding transition. These results were successful only in certain limited conditions for 1GB1 because it contains both α-helix and several β-sheets. Such proteins with α- β content have not been studied vigorously. The other two proteins yielded satisfactory results under all conditions, and this was expected based on prior research for such α-helical proteins. Further testing would be required to validate the application of this method to α- β content proteins. The folding trajectories of 1YRF and 1L2Y were easily replicated under various simulation conditions, whereas results for 1GB1 were minimal. We need additional validation for α- β content, and until we do so it is uncertain whether the GPA* algorithm should be applied to larger proteins.

Chapter 3 Methodology

We picked up where Syzonenko & Phillips (2020) left their research, by studying and continuing to improve their methodology. We developed our code in Python for better portability. We also used a shelve storage system, which is database inspired. Figure 4 describes an example of how each node is stored in the shelve data structure.

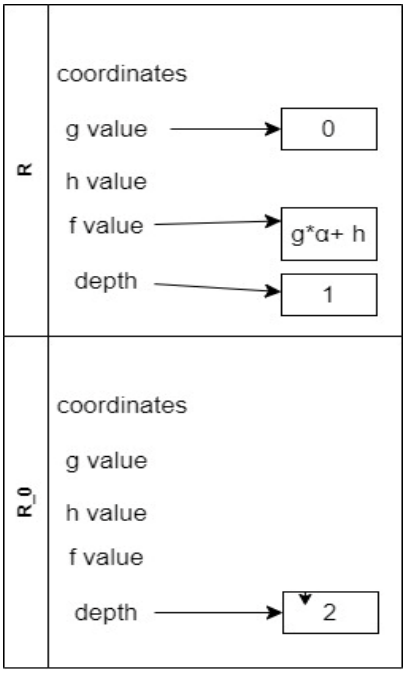

Figure 4 Key-Value Data Structure for Storage of Generated Search Tree

We used the Worldwide Protein Data Bank (wwPDB.org, 2021), which is an archive of experimentally determined 3D structures, to obtain the protein structures and conformations of 1GB1 (Gronenborn et al., 1991).

We used the common tools used in molecular dynamics computing, like GROMACS, to run our simulations (Lemkul, J. A. 2018), VMD which is a popular 3-D modeling tool (Humphrey et al., 1996), and the BioSim Computer Cluster, available via the MTSU (Middle Tennessee State University) Computer Science department, to run and parallelize our biomolecular simulations.

We also utilized MDAnalysis, which is a toolkit commonly used for analyzing MD simulations (Michaud-Agrawal et al., 2011). GROMACS is a free and open-source software that is commonly used for high-performance molecular dynamics and output analysis. It follows a modular process, which means it works in stages like pre-processing and post-processing to make the processes more efficient. We follow a similar approach at the core of our implementation.

The streptococcal protein G (1GB1), is a fast-folding protein and has an α-helix and several β-sheets, making it difficult to predict how 1GB1 folds due to the complex energy landscape shown by such α-β proteins. Because of these properties, it is commonly used to validate MD sim.

3.1 Obtaining the Unfolded Structure for 1GB1

We ran the NVT simulations for 10 ns at 600K to obtain the unfolded structure for 1GB1, which is what we use as the initial structure to start the search process. We started with the PDB (Protein Data Bank) structure from (Gronenborn et al., 1991) The steps followed for system preparation can be found here: http://www.mdtutorials.com/gmx/lysozyme/index.html.

In the above simulation, we kept pressure-coupling and position-restraints off during the production stage. No pressure coupling means that there is a fixed box size for the atoms in the system. Position restraints are used during equilibration to avoid drastic rearrangements of critical parts of the protein (*User guide#*). We used the AMBER94 (Lindorff-Larsen et al., Proteins 78, 1950-58, 2010) force field with the recommended T1P3P water mode.

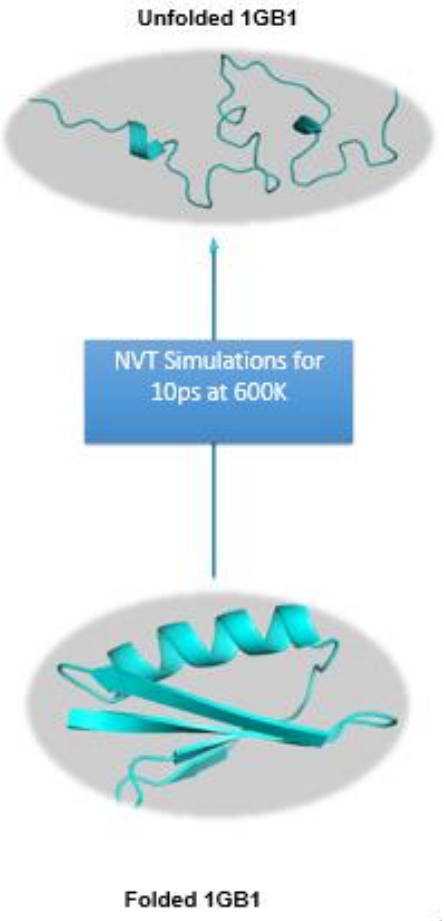

Figure 5 Obtaining the Unfolded 1GB1 Structure

3.2 Code Implementation

The previous implementation of GPA* has certain limitations and benefits. Their implementation was done using the language C++. Python (programming language) is slower than C++ because C++ is statically typed, which speeds up code execution. However, Python improves readability and can be easier to understand for biologists not familiar with C++. We reimplemented our code to use python. Our re-implementation can be found here: https://github.com/For-am/GPAstar/tree/UndergradThesis

We are re-implementing the existing code. They stored all files on disk, causing billions of auxiliary files, which became difficult to work with. As we now know that the algorithm has major success, we need not worry about the auxiliary files. As they were still in the preliminary stages of evaluating the algorithm, they needed to have all the files available for debugging purposes. They were focused on getting the algorithm to work and evaluating its accuracy. Now that we know that the algorithm works, we can focus more on reconstructing the code and modifying the tools used by the code to make it more user-friendly.

Furthermore, the code is limited in portability and does not have modularity. It also has scalability issues on typical file systems used nowadays. Current storage mechanisms are usually database systems instead of using on-disk storage of files.

Their code was a single code block implemented for all steps. By breaking the process up into phases, our code fragments can work independently. This is like the modular phase system GROMACS also uses. We divided our code into three phases: Pre-Processing, Search process, and post-Processing.

All files are easily accessible to all three of the code sections, as they all use a common shelve data structure. We used MDAnalysis to save disk storage space and avoided having thousands of gro files. Ongoing research aims to accelerate these Molecular Dynamics (MD) simulations.

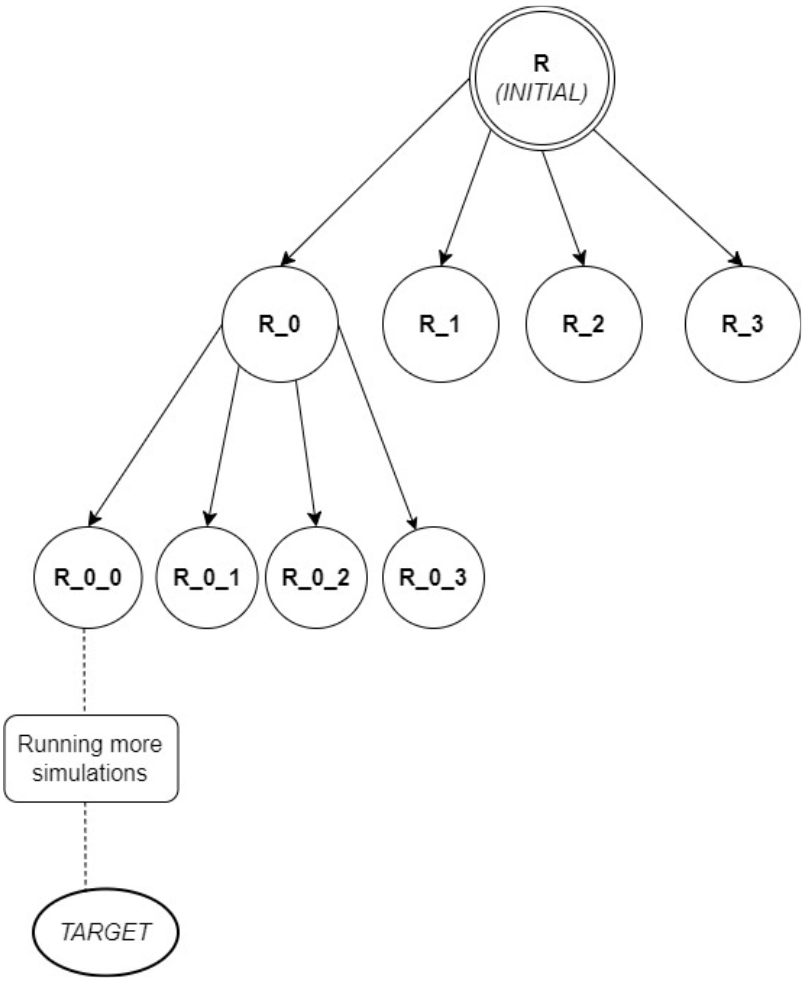

Figure 6 Generating a Search Space

3.2.1 Code Implementation

In the preliminary MD simulations phase, we generated a search space, which is a graph/tree of the possible protein structures generated. According to the user-provided seed value N, the code generates N number of mdp files with n seeds. Then the initial coordinates are stored in the shelve, and we initialize all values in the shelve to get started. In Figure 6, we can see how there is a root node 'R', we initialized this node here.

The user only provides the following files to get started: mdp file, topology file, initial and target gro file; the rest is handled by the code:

1. The topology file is built following the GROMACS specification for a molecular topology. A topology file can be generated by pdb2gmx, which is a commonly used GROMACS command (*User guide#*).
2. The GRO file format contains information about the positions of all atoms in a molecular system, as well as the box size and shape, and the velocity of each atom if available. GROMACS uses the GRO file format as an input and output file format for many of its tools and utilities, such as energy minimization, equilibration, and molecular dynamics simulations. GRO files can be easily viewed and edited using text editors or molecular visualization software, such as VMD. (*User guide#*) The initial gro file is the unfolded protein structure we start with, and the target gro file is the final folded structure we would like to achieve.
3. In GROMACS, an MDP file is a text file that contains all the input options and settings for a molecular dynamics simulation (Berendsen, van der Spoel, & van

Drunen, 2015). The only settings we overwrite to the user-provided mdp file are setting gen_vel to ‘yes’ and changing the seed values while generating the mdp files needed. The seed values simply ensure that whenever the simulation is rerun, the same output is obtained. Setting gen_vel to ‘yes’ ensures that velocities are generated according to a Maxwell distribution at given temperature gen-temp, with provided seed values (*User guide#*). The user has the flexibility to customize the simulation as they want. MDP stands for "Molecular Dynamics Parameter" file, and it contains information on the simulation conditions such as temperature, pressure, time step, integrator algorithm, and force field parameters. It allows the user to customize and optimize the simulation parameters for their specific research needs. The file is highly customizable, and GROMACS provides a comprehensive manual to help users understand and modify the different options available (Berendsen, van der Spoel, & van Drunen, 2015).

Chapter 4 Results

Using Python instead of C++ makes our code implementation quicker. We have been working on code development and figuring out methods to make the entire process scalable. The initial barrier we had was that the number of files generated per simulation was so much that it became burdensome on the file systems. We used a shelve, which is a key-value data structure, for storage instead of storing auxiliary files on disk. Instead of relying on GROMACS commands using OS system calls, we switched to using the MDAnalysis library, which is based on GROMACS at the core for simulation commands and storing coordinates. We currently have an implementation with GROMACS successfully running without errors and can now be made more efficient using the MDAnalysis library. Our preprocessor implementation enables the user to customize the simulation runs according to their requirements, as they only must provide us with the three files: initial structure, target structure, and required mdp files. The user has the flexibility to customize the simulation settings as they wish.

Chapter 5 Future Work

It is recommended to use the following two stages, after using our preprocessor code, when running the simulations and obtaining results from the code:

5.1 GPA Search Code

The first stage is where we would run the actual simulations and generate more nodes as we progress through the search process. The main loop must be structured in such a way that fault recovery is possible, as the loop runs the net simulations only for keys not present in the tree. We will end up with a final key that describes the best sequence of keys used to build the complete pathway.

5.2 Post-Processing

The second stage is the compilation of all the data gathered and this part of the code is under development. We decided to store only the atoms of each gro file in our shelve data structure and discard all auxiliary files, as there were many files generated in each run, and storage issues must be mitigated to proceed. From the key we obtain in the Search phase, we can now rerun the simulations in the order specified and then use the files obtained to view the complete pathway using a 3-D modeling tool like VMD.

Chapter 6 Conclusion

We expected the approach to yield positive results, as the former researcher found success with other protein structures. This method also removes the unnatural biases we discussed earlier. In summary, this thesis is a preliminary step in the validation of the greedy-proximal A* algorithm on the complex and fast-folding 1GB1 protein.

Once this is validated, the approach may be applied to even larger proteins and across a broader spectrum of biological systems. Proteins with α- β content have not been studied vigorously. Further testing would be required to validate the application of this method to α- β content proteins. The GPA* algorithm is expected to lower the computing time required to acquire the folded conformation without adding artificial energy bias and to enable trajectory design with minimum movements required for the folding transition.

References

Ben-Naim, A. (2012). Levinthal's question was revisited and answered. *Journal of Biomolecular Structure and Dynamics*, *30*(1), 113–124. https://doi.org/10.1080/07391102.2012.674286

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT Press.

Geiler, K. (2014, January 9). *Protein folding: The good, the bad, and the ugly*. Science in the News. Retrieved January 19, 2022, from https://sitn.hms.harvard.edu/flash/2010/issue65/

Gershenson, A., Gosavi, S., Faccioli, P., & Wintrode, P. L. (2020). Successes and challenges in simulating the folding of large proteins. *Journal of Biological Chemistry*, *295*(1), 15–33. https://doi.org/10.1074/jbc.rev119.006794

Gronenborn, A. M., Filpula, D. R., Essig, N. Z., Achari, A., Whitlow, M., Wingfield, P. T., & Clore, G. M. (1991). A Novel, Highly Stable Fold of the Immunoglobulin Binding Domain of Streptococcal Protein G. *Science*, *253*(5020), 657–661. https://doi.org/10.1126/science.1871600

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics, 4(2), 100-107.

Humphrey, W., Dalke, A., & Schulten, K. (1996). VMD: Visual molecular dynamics. *Journal of Molecular Graphics*, *14*(1), 33–38. https://doi.org/10.1016/0263-7855(96)00018-5

Lemkul, J. A. (2018.). *Lysozyme in Water*. http://www.mdtutorials.com/gmx/lysozyme/index.html

Lombardi, A., & Nigro, F. (2018). Metadynamics: Theory, Implementation, and Applications. In Reference Module in Life Sciences. Elsevier. https://doi.org/10.1016/B978-0-12-809633-8.20284-7

McGill University. (2011, June 8). *Protein folding made easy*. ScienceDaily. Retrieved January 19, 2022, from https://www.sciencedaily.com/releases/2011/06/110607121135.htm

Michaud-Agrawal, N., Denning, E. J., Woolf, T. B., & Beckstein, O. (2011). MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations. Journal of computational chemistry, 32(10), 2319. https://doi.org/10.1002/jcc.21787

Pfaendtner, J. (2019). Metadynamics to Enhance Sampling in Biomolecular Simulations. *Methods in Molecular Biology*, 179–200. https://doi.org/10.1007/978-1-4939-9608-7_8

Qi, R., Wei, G., Ma, B., & Nussinov, R. (2017). Replica Exchange Molecular Dynamics: A Practical Application Protocol with Solutions to Common Problems and a

Peptide Aggregation and Self-Assembly Example. *Methods in molecular biology (Clifton, N.J.)*, *1777*, 101. https://doi.org/10.1007/978-1-4939-7811-3_5

Russell, S. J., & Norvig, P. (2010). Artificial intelligence: a modern approach (3rd ed.). Prentice Hall.

Sharma, S. K., & Kumar, S. (2016). COMPARATIVE ANALYSIS OF MANHATTAN ANDEUCLIDEAN DISTANCE METRICS USING A* ALGORITHM. *Journal of Research in Engineering and Applied Sciences*, *01*(04), 196–198. https://doi.org/10.46565/jreas.2016.v01i04.007

Syzonenko, I., & Phillips, J. L. (2020). Accelerated Protein Folding Using Greedy-Proximal A*. *Journal of Chemical Information and Modeling*, *60*(6), 3093–3104. https://doi.org/10.1021/acs.jcim.9b01194

Toews, R. (2021, December 10). AlphaFold Is The Most Important Achievement In AI—Ever. *Forbes*. https://www.forbes.com/sites/robtoews/2021/10/03/alphafold-is-the-most-important-achievement-in-ai-ever/?sh=5d893fb26e0a

User guide#. User guide - GROMACS 2023 documentation. (n.d.). Retrieved March 10, 2023, from https://manual.gromacs.org/documentation/current/user-guide/index.html.

wwPDB.org. (2021). *wwPDB: Worldwide Protein Data Bank*. Wwpdb.org.

Zhou, R. (2022). Replica Exchange Molecular Dynamics Method for Protein Folding Simulation. *Protein Folding Protocols*, 205–224. https://doi.org/10.1385/1-59745-189-4:205